'Rough Enough' - A System Supporting the Rough Sets Approach

Anders Torvill Bjorvand
Troll Data Inc.
P.O. Box 335, N-1801 Askim, Norway
Phone +47 69 88 99 70, Email torvill@trolldata.no

Abstract

The rough sets theory has proved useful for a wide variety of data mining tasks. A system, called 'Rough Enough', has been developed to support the basic operations of the theory. The system emphasizes modifiability and experimentalism. Genetic algorithms have been used for the NP-hard rough sets procedure of data reduction - obtaining the set of reducts. The system has been successfully tested with real world data.

1 Introduction

The 'Rough Enough' system was initiated by the author at the Norwegian Institute of Technology (now: the Norwegian University of Science and Technology) under the supervision of Professor Jan Komorowski.

The major goal of the system was to support the basic operations of the rough sets theory. An important design issue was to obtain a system which was easy to modify. This was to support easy experimentation with diverse techniques and algorithms within the field of rough sets. Different genetic algorithms have been used for the NP-hard rough sets procedure of data reduction - i.e. to obtain the set of reducts. The system has been shown to work on real world data sets containing echocardiogram measurements and stock market data.

2 Rough Sets Theory

Rough sets theory (from now on called RS) is a mathematical tool for approximate reasoning for decision support. It was introduced by Zdzisław Pawlak in the early eighties and is thoroughly documented in his book from 1991, [8]. RS is particularly well suited for classification of objects. In a wider perspective, the theory belongs to the new branch of research in data mining and knowledge discovery. A very brief

introduction to RS will be given here, but a wider presentation can be found in eg [8].

RS deals with different levels of granularity in information. The amount of information you have about an object limits your ability to classify/recognize the object.

In traditional set theory, an object is either in the set or not in the set. In RS, however, an object can also be in a third state: possibly in the set. In the following presentation, I will rely on much of the material in [9], [8], [1], [2], [4], [3] and [5].

2.1 Information Systems

RS relies on information systems for the representation of data. The information system is formalized as follows:

Definition 1 ***Information System*** $\mathbf{A} = (U, A)$
U - *set of objects(cases,states,patients,observations,...)*
A - *set of attributes (features,variables,characteristic conditions,...)*
$U \neq \emptyset$, $A \neq \emptyset$
$a : U \longrightarrow V_a$, $a \in A$
V_a – *value set*

Comment An information system is a set of objects described by attributes.

Definition 2 ***Decision table*** - *this is in many ways an extension/specialization of an information system. It is therefore customary to use the same symbol,* **A**, *to represent it:*
$\mathbf{A} = (U, A \cup \{d\})$
$d \notin A$

Comment A decision table is an information system with an extra decision attribute.

2.2 Set Approximations

Definition 3 ***Indiscernibility***
With every subset of attributes $B \subseteq A$ *we associate a binary relation* $IND(B)$, *called* B – *indiscernibility relation , which we define as follows:*

$$IND(B) = \left\{(x, y) \in U^2 : \text{ for every } a \in B,\ a(x) = a(y)\right\}$$

if $(x, y) \in IND(B)$, *we say* $xIND(B)y$ (x *and* y *are indiscernible by attributes* B)

$IND(B)$ *divides the universe,* U, *into equivalence classes,*

$$[x_i]_B = \{x_j \in U : \ x_i IND(B) x_j\} \tag{1}$$

Two objects are indiscernible with respect to their descriptive attributes if all their attribute values are equal, i.e. they appear the same. These equivalence classes contain all objects which are indiscernible from each other.

Definition 4 ***Set approximations***

$$\underline{B}X = \{x \in U : [x]_B \subseteq X\}$$
$$\overline{B}X = \{x \in U : [x]_B \cap X \neq \emptyset\}$$
$$BN_B(X) = \overline{B}X - \underline{B}X$$

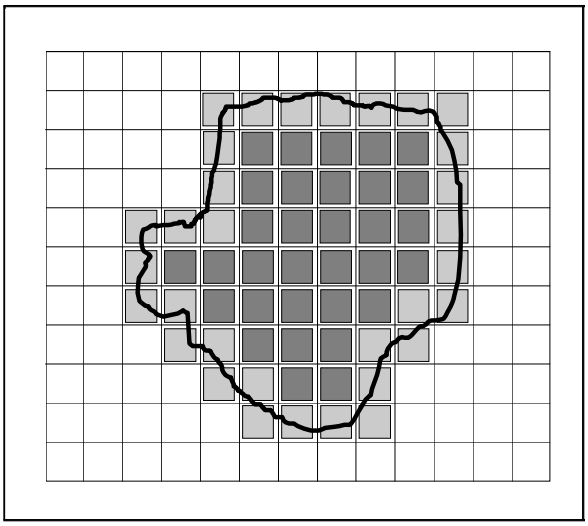

Figure 1: Simple illustration of roughness.

$B \subseteq A$
$X \subseteq U$
$\underline{B}X$: *B-lower approximation of X*
$\overline{B}X$: *B-upper approximation of X*
$BN_B(X)$: *B-boundary region of X.*

Comment I will use figure 1 to explain. The squares are the equivalence classes of the IS with respect to B as defined in equation 1. The area closed by the freehand drawing is X. The B-lower approximation is then the dark gray squares. The B-upper approximation is then both the dark and the light gray squares. The B-boundary region is the light gray squares. Elements of the dark gray squares are definitely within the area of the drawing, elements of the light gray squares are possibly within the area of the drawing.

2.3 Data Reduction

The discernibility matrix was introduced by Andrzej Skowron, and made one of its first appearances in 1991 in a predecessor to his 1992 paper: [9]. The discernibility matrix is a structure used for the data reduction.

Definition 5 ***Discernibility matrix*** **-** *M*
$M(\mathbf{A})$ is the $n \times n$ discernibility matrix of the information system $\mathbf{A}$ with coefficients c_{ij}, such that:

$$c_{ij} = \{a \in A : \quad a(x_i) \neq a(x_j)\}$$
$$i, j \in [1..n]$$
$$n = |\mathbf{A}| \, (i.e. num. of attributes)$$

We have one index in the discernibility matrix for every combination of two objects from the information system. This index contains the list of individual attributes for which these two objects have different values. Each attribute on this list can therefore discern between these two objects.

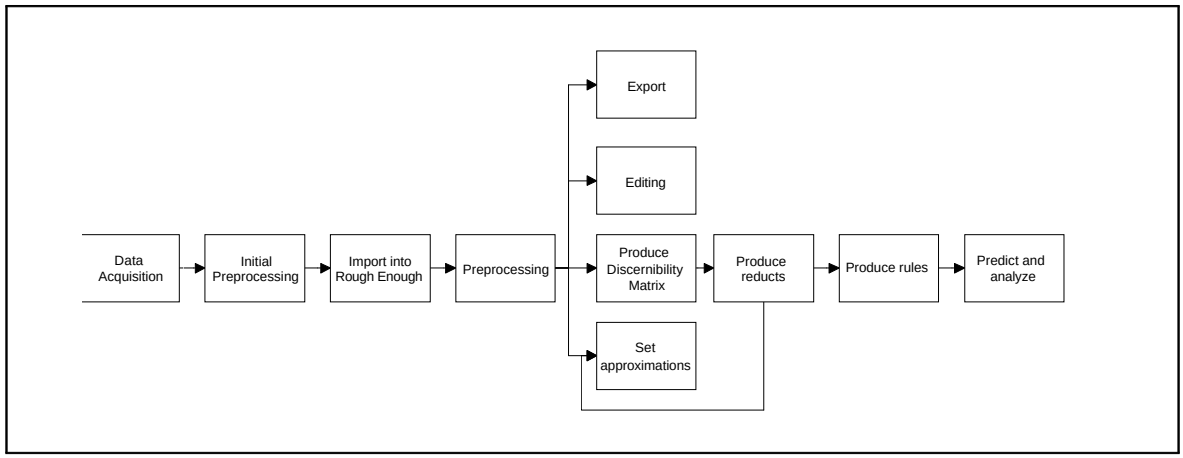

Figure 2: Illustration of the 'Rough Enough' workflow

Definition 6 ***Reducts***
Any minimal $B \subseteq A$ *such that* $IND(A) = IND(B)$ *is a reduct in information system* **A**.
$RED(\mathbf{A})$ *- the set of all reducts for* **A**.
With minimal, we mean that if any one attribute is removed from the reduct, it will no longer be able to distinguish between all objects.

Comment Reducts are constructed to get rid of superfluous attributes. In that way, you obtain the following: fewer attributes give faster computations, and knowing which attributes are superfluous can save you from much future measurement work.

Definition 7 ***Core***

$$CORE(\mathbf{A}) = \bigcap RED(\mathbf{A})$$

3 The Architecture and Design of 'Rough Enough'

In this section, I will explain the basic architecture of 'Rough Enough'.

3.1 Workflow

The intended workflow for working with 'Rough Enough' is illustrated in figure 2.
In the further explanations I will rely on this workflow to explain both the functionality and the architecture. The two first points, data acquisition and initial preprocessing, are not supported in 'Rough Enough', and will therefore not be covered.

3.2 Import into Rough Enough

The system supports import from most PC database and spreadsheet formats. The Rough Set Expert System (RSES) (see [11]) is also supported. SQL servers are easily accessible by just switching the database drivers. This makes the jump to client/server and the use of corporate data an easy one.

3.3 Preprocessing

It is very important to use a relevant preprocessing of the data to achieve the data

representation relevant for what you want to do. 'Rough Enough' has several methods of preprocessing. You may choose from the following methods:

- Absolute change - this method calculates the difference between the current object and the previous with respect to the object numbering. See [2] for further discussions of this technique.

- Percent change - this method calculates the difference between the current object and the previous object in terms of a percent change.

- Split table - this method splits your table in two based on user input about how many objects you want to keep. The removed objects will be stored elsewhere and used to test rules applied to these unseen objects (cf. section 3.8).

- TIS to IS - translates a temporal information system into an information system. Temporal information systems represent time series. The algorithm for this procedure and the introduction to temporal information systems can be found in [2] and [3].

In addition to these predefined techniques, the user has direct access to SQL and QBE. This is very convenient if we want to carry out a special task (which is often the case when it comes to data mining).

3.4 Editing Data

Data can be readily edited in the 'Rough Enough' system. Data can be edited on the main screen of the system as shown in figure 3.

3.4.1 Working With a Subset

You can choose to work with a subset of the attributes. This is done by clicking on the titles of the attributes. You can automatically highlight all attributes that are members of a certain stored reduct. These reducts are the ones found and stored using the techniques of section 3. One example where attributes 1 through 6 except 5 are highlighted can be seen in figure 4.
This facilitates experimentation with reducts to find the quality of set approximations and rules since the production of rules and set approximations rely on this attribute selection.

3.5 Produce Discernibility Matrix

The discernibility matrix is used for the calculation of reducts and cores. We are actually working with a slight variation called a distinction table which will be presented in section 3.7. The discernibility matrix will be computed according to which attributes are chosen/highlighted. The discernibility matrix screen is shown in figure 5.

3.6 Set Approximations

The system has several facilities to work with set approximations: equivalence classes, decision classes, lower approximation, upper approximation, boundary region, rough membership value and generalized decision rule. All these computations again rely on

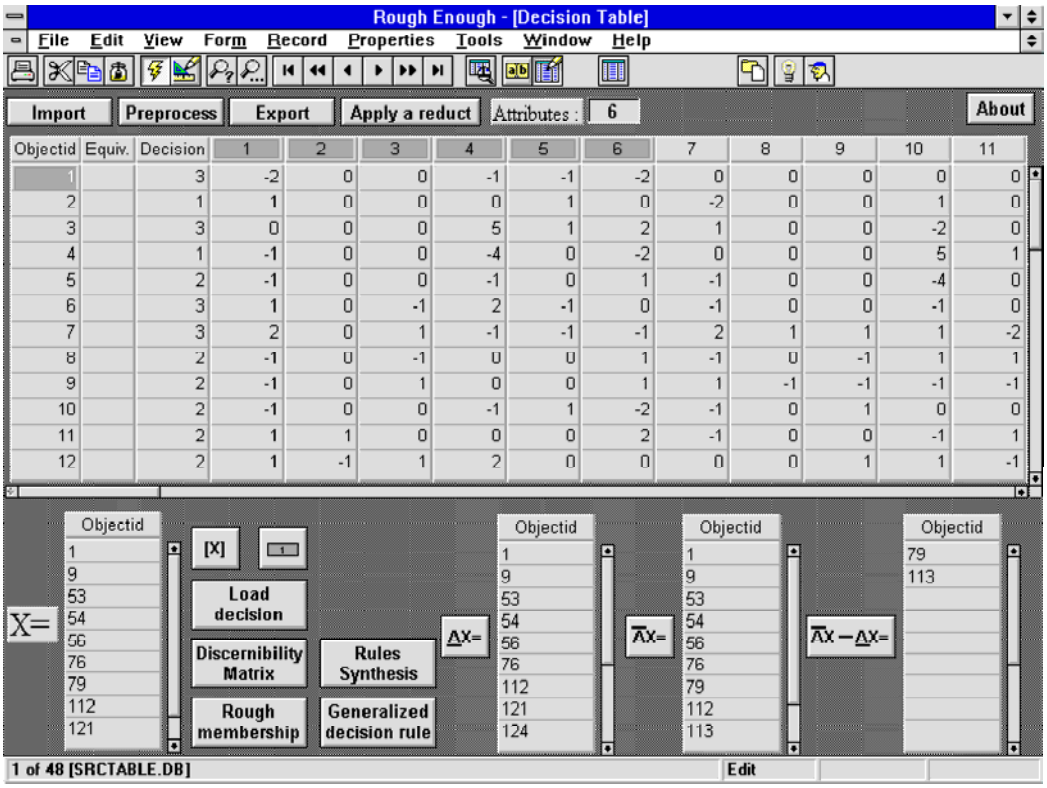

Figure 3: The main screen of 'Rough Enough'.

the selections of attributes that you have chosen. These operations are rather trivial, so I will not elaborate on them any further. See [2] for further details.

3.7 Produce Reducts

The algorithms for calculating reducts will now be presented. But some preliminaries are necessary:
The distinction table is a special representation of the discernibility matrix. The representation was introduced in [12]. The representation is very efficient for reduct-finding with a computer.

Definition 8 ***Distinction Table***
Let B be a binary matrix with dimensions $(N+1) \times \frac{m^2-m}{2}$ where N is the number of attributes and m is the number of objects of our decision table. The N+1'st attribute being the decision. Let $b(i,(k,n))$ be an element of B corresponding to the attribute i and pair of objects (x_k, x_n) :

$$b(i,(k,n)) = \begin{cases} 1 & if \quad a_i(x_k) \neq a_i(x_n) \\ & \quad for\ i \in \{1,...,N\} \\ 0 & if \quad a_i(x_k) = a_i(x_n) \end{cases}$$

$$b(n+1,(k,n)) = \begin{cases} 0 & if \quad d_i(x_k) \neq d_i(x_n) \\ \\ 1 & if \quad d_i(x_k) = d_i(x_n) \end{cases}$$

In terms of matrix B, to find the reduct means to find the minimal subset of columns

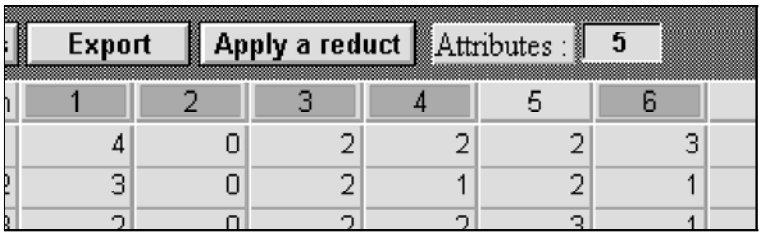

Figure 4: The appropriate attributes of the chosen reduct are highlighted.

$R \subseteq \{1, ..., N\}$ *satisfying:*

$$\forall (k,n) \exists i \in R : b(i,(k,n)) = 1 \ \vee \ b(N+1,(k,n)) = 1 \tag{2}$$

In practice, an even better way of doing this is to remove all rows of the distinction table where the decisions are equal i.e. $b(N+1,(k,n)) = 1$.
This representation can very easily be given a binary representation in a computer, utilizing fast bitwise logical processor instructions.

By removing duplicate rows of the distinction table, we can improve time/space efficiency even further.

3.7.1 Exact Solution

A very obvious way to obtain the reducts is to try all combinations of attributes starting with all candidates with zero attributes, go on to all candidates with one attribute, two attributes and so on. The first candidate which satisfies formula 2 belongs to the set of globally minimal reducts. Going on from here, the rest of the reducts are found in the same way, but we have to check if the attributes of our candidates are supersets of the reducts that we have already found. If they are, they are not reducts.
By showing polynomial reducibility from a problem which has been shown to be NP-hard to this problem, it was shown in [9, pp.343-344] that the minimal reduct problem, and hence the problem of finding all reducts, belongs to the class of NP-hard problems. We need some method of approximation to solve our problem with reasonable resources. The successful application of genetic algorithms has been reported in [12], and we chose this approximation method for our system. Our solution is also related to one of the solutions presented in [12].
To give an overview of genetic algorithms here would be beyond the scope of this paper. A general understanding of the workings of genetic algorithms is required for the following section. A good introduction can be found in [7].

3.7.2 Approximate Solutions Using Genetic Algorithms - Preliminaries

Representation I represent my chromosomes (candidate solutions) as bit strings of length N, where N is the number of attributes. 1 means that an attribute is present, and 0 means that it is not.

Example 1 *Assume that we have 10 attributes* $\{a_1, a_2, ..., a_{10}\}$ *and that we have the following reduct candidate:* $\{a_3, a_7, a_9, a_{10}\}$. *This should be represented as:* $\vec{v} =$ 0010001011.

The approximations of the reducts are carried out with genetic algorithms, and I have implemented two strategies.

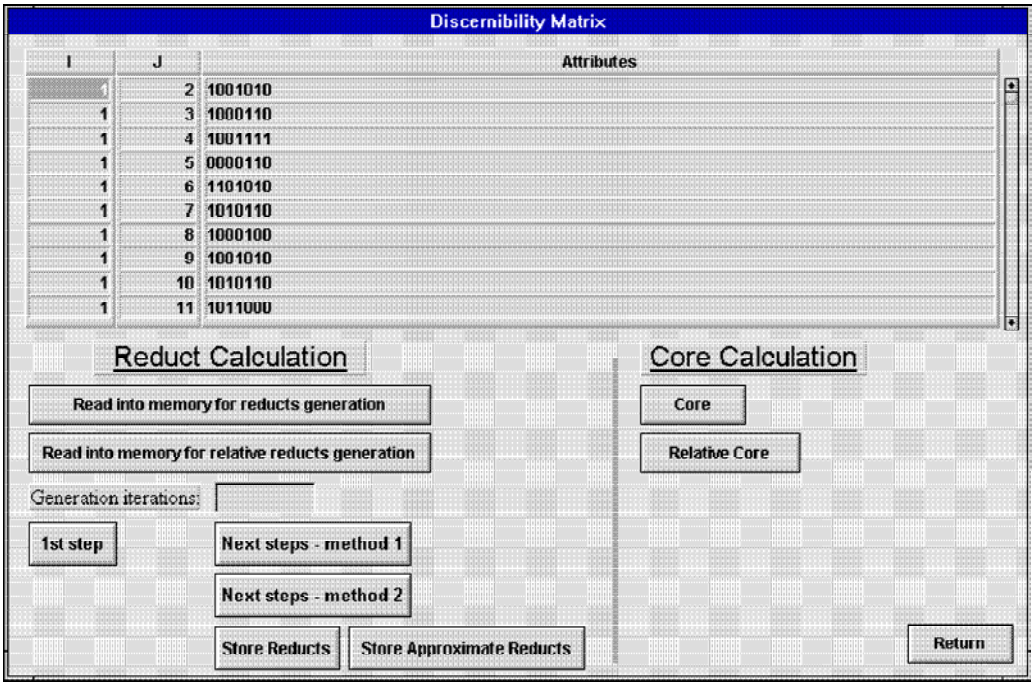

Figure 5: This is the screen for the computation of reducts and cores.

Strategy 1 The fitness function:

$$F(\vec{v}) = \begin{cases} \left(\frac{N-L_{\vec{v}}}{N} + \frac{C_{\vec{v}}}{(m^2-m)/2}\right)^2 & if \quad C_{\vec{v}} < (m^2-m)/2 \\ \left(\frac{N-L_{\vec{v}}}{N} + \left(\frac{C_{\vec{v}}}{(m^2-m)/2} + \frac{1}{2}\right) \cdot \frac{1}{2}\right)^2 & if \quad C_{\vec{v}} = (m^2-m)/2 \end{cases}$$

Where $L_{\vec{v}}$ is the number of occurrences of 1 in $\vec{v}$
and $C_{\vec{v}}$ is the number of object pairs that $\vec{v}$ can discern between)

We square the fitness function to better separate the different values. The part with the coverings is also multiplied with $\frac{1}{2}$ so as to avoid that reducts (ie with complete covering) are given a low fitness value compared to other candidates that are close to a reduct but have a much smaller number of attributes.
We set the mutation rate according to the redundancy of our individuals. If we have many equal individuals, we set the mutation rate higher. We also use a higher probability of mutating from a 1 to a 0 than for the opposite direction, since our goal is short reducts.

Strategy 2 This strategy is equal to the previous one, but we store the individuals in a temporary storage before we mutate them. After mutation, we select the best individuals from those which have not mutated, and those which have mutated.
This strategy has found reducts faster than strategy 1, but every generation takes twice as long to calculate, since we are evaluating the individuals both before and after the mutation. This method also seems to get stuck in one search direction.
The most efficient way to find reducts has been to combine the two strategies: using strategy 2 for fast reduct finding, and strategy 1 for finding new and interesting paths of further search for reducts. Strategy 2 was used to find reducts in market data. The globally minimal reduct (6 attributes) was found after 26 generations. The original

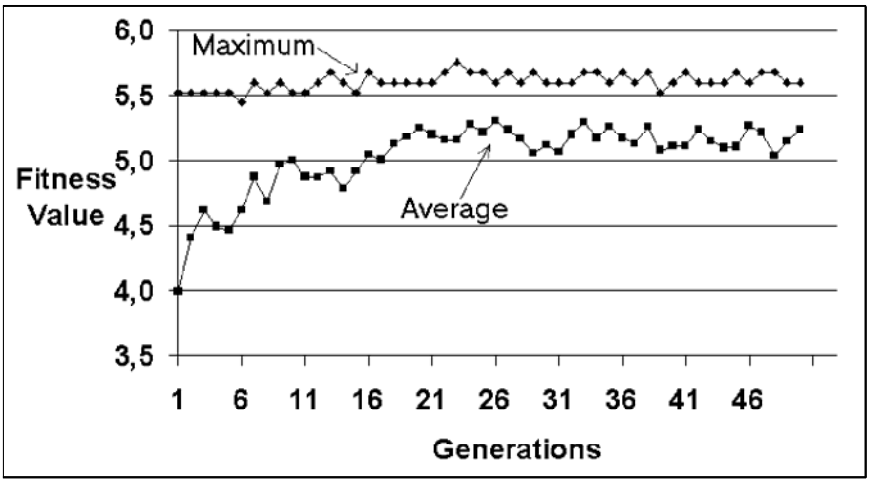

Figure 6: The development of the reduct approximation.

number of attributes was 60. The development of this experiment can be seen in figure 6.

3.7.3 Storing Reducts

There are two versions of this routine, one absolute and one approximate. The absolute version searches for the globally minimal reduct. The approximate solution searches for candidates that discern between sufficiently many objects according to a limit specified by the user. These "reducts" are often better than the true reducts since they are not so vulnerable to the occurrence of a few noisy objects. Only candidates with a lower number of attributes than the globally minimal reduct will be stored. The search is performed in the database of chromosomes accumulated during the generations of our genetic algorithms.

3.8 Produce Rules, Predict and Analyze

The normal procedure for working with rules in 'Rough Enough' is shown in figure 7.

3.8.1 Split Data Into Training Set and Test Set

This is done in the preprocessing part of the system. The user specifies the percentage of objects that should be in respectively the training set and the test set. The training set is used for producing rules, and the test set is used to evaluate the rules.

3.8.2 Create Rule Candidates

In this step, all possible and impossible rules based on a complete cartesian product of the valuesets of our attributes are generated.

3.8.3 Evaluate Rule Candidates

This step of the process will assign each rule a positive vote for each object supporting it, and a negative vote for each object that opposes the rule.

3.8.4 Remove Rules That Are Not Supported

After the generation process, some criteria can be set to remove some obviously unnecessary rules.

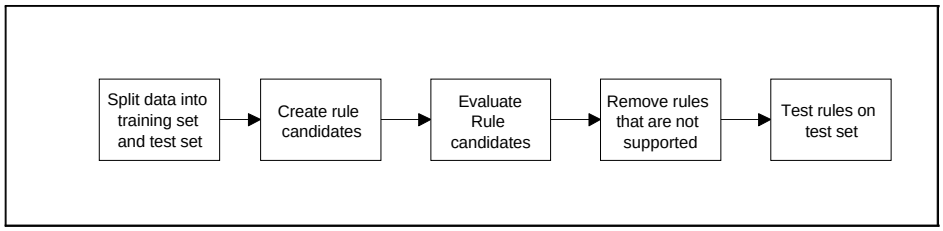

Figure 7: The normal procedure for producing and testing rules

- Remove rules that do not have at least one object in the training set supporting it.

- Remove non-supported rules - this function removes all rules which are opposed by more objects than are in favour of them.

3.8.5 Test Rules On Test Set

This is the step of the process where the decisions of the test set are predicted. Several strategies/partial strategies are implemented here:

- Strategy 1 - this method chooses the matching rule with the biggest positive difference between the positive and negative votes.

- Strategy 2 - this method will first consider the exact rules (with no negative votes) and then the inexact rules (with some negative votes). In this strategy, the rule with the highest product obtained from multiplying the length of the rule with the difference of the positive and negative votes will be chosen. This gives preference to the longer, less general rules.

- Strategy 3 - for each object we construct the set of all the longest rules that match. We choose the single rule with the highest positive difference between the positive and the negative vote.

- Apply random - this method assigns random decisions to the test set where the probability of assigning each decision value is proportional to the frequency with which it occurs in the learning set. This technique is used to evaluate our three rule strategies to ensure that the results are better than pure guessing.

The system will report the number of correct and incorrect predictions with the appropriate percentages. This will of course demand that we know the real decision values. The user is also able to inspect the predictions side by side with the correct values for each object in the test set.

3.9 Export of Data

The system supports export to most PC database and spreadsheet formats. The Rough Set Expert System (RSES) is also supported.

4 Applications of the System

It would be beyond the scope of this article to go into the applications of the system in any depth. The system has however been tested on echocardiogram data and stock market data. The system was able to show logical relations in both data sets. The data reduction was most successful with the stock market data, where we reduced the number of attributes from 60 to 6. Detailed descriptions of these experiments can be found in [2].

5 Related Research

Many have implemented software to support reasoning based on the rough set approach. They all have one thing in common - they have been developed by researchers and research groups with many new ideas and suggestions for extensions of the rough set approach itself, extensions of both formalisms and most of all different algorithms with more or less the same goals. As one would expect, the respective software packages reflect the research carried out by its developers. Since RS is a relatively young theory, this broad diversity should be expected. They all have slightly different theoretical emphasis, and in many respects they have been developed to aid the respective researchers in carrying out their specific experiments.
Some of the better known systems are Datalogic/R [13], LERS [6], RoughDAS / Roughclass [10] and RSES [11].

6 Conclusion

A system has been developed. The system runs on PCs with the operating system Microsoft Windows/DOS. The system has proved to work with experimental data. Experiments have been carried out with new variations of genetic algorithms for reduct finding. The reducts are found in a far more effective way than any deterministic solution.
The main difference of focus in 'Rough Enough' as opposed to other existing systems is the emphasis on easy modifiability. The aim has been to let other researchers share my experiences and even collaborate on the system.
The system is also well suited for learners of the rough set approach. Since most introductions to RS have been rather theoretical, the theory might be a little hard to adapt to real world situations at first. By implementing small examples and testing in 'Rough Enough', you can easily verify and broaden your understanding as you learn.
'Rough Enough' can be downloaded from the following Internet-URL:
http://home.sn.no/~torvill.

References

[1] Bjorvand, Anders Torvill (1995). *Fractal Logic - A New Approach to Intelligent Systems Theory.* Project report - Course 45073: Computer Science Projects. The Norwegian Institute of Technology. Trondheim, Norway.

[2] Bjorvand, Anders Torvill (1996). *Time Series and Rough Sets.* Master's Thesis - The Norwegian Institute of Technology (now: the Norwegian University of Science and Technology). Trondheim, Norway.

[3] Bjorvand, Anders Torvill (1997). *Mining Time Series Using Rough Sets - A Case Study.* PKDD'97 - 1st European Symposium on Principles of Data Mining and Knowledge Discovery. Trondheim, Norway. Springer Verlag Lecture Notes in Computer Science.

[4] Bjorvand, Anders Torvill (1997). *Rough Enough - Software Demonstration.* 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics - to appear.

[5] Bjorvand, Anders Torvill and Komorowski, Jan (1997). *Practical Applications of Genetic Algorithms for Efficient Reduct Computation.* 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics - to appear.

[6] Grzymała-Busse, J. W. and Sikora D. J. (1988). *LERS1 - A System for Learning from Examples Based on Rough Sets.* Report TR-88-5, Department of Computer Science, University of Kansas.

[7] Michalewicz, Zbigniew (1995). "Evolutionary Computation: An Overview". Printed in *The Proceedings of SCAI'95*. Edited by A. Aamodt and J. Komorowski. IOS Press.

[8] Pawlak, Zdzisław (1991). *Rough Sets - Theoretical Aspects of Reasoning About Data.* Kluwer Academic Publishers, Dordrecht.

[9] Skowron, Andrzej and Rauszer, Cecylia (1992). "The Discernibility Matrices and Functions in Information Systems". Printed in *Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory.* Edited by Roman Słowiński. Pp. 331-362. Kluwer Academic Publishers, Dordrecht.

[10] Słowiński, Roman and Stefanowski, Jerzy (1992). "'RoughDas' and 'RoughClass' Software Implementations of the Rough Sets Approach". Printed in *Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory.* Edited by Roman Słowiński. Pp. 331-362. Kluwer Academic Publishers, Dordrecht.

[11] Synak, Piotr (1995). *Rough Set Expert System User's Guide - Version 1.0.* Developed by the group supervised by Professor Andrzej Skowron at the Institute of Mathematics, the University of Warsaw.

[12] Wróblewski, Jakub (1995). *Finding Minimal Reducts Using Genetic Algorithm (extended version).* Warsaw University of Technology - Institute of Computer Science - Reports - 16/95.

[13] Ziarko, W., Golan, R. and Edwards, D. (1993). *An application of Datalogic/R Knowledge Discovery Tool to Identify Strong Predictive Rules In Stock Market Data.* AAAI-93 Workshop on Knowledge Discovery in Databases.